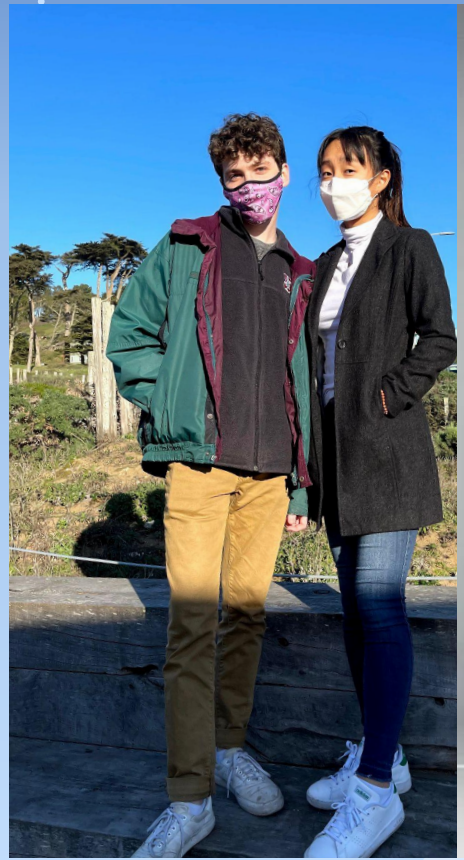
YEAH: Getting Your C++ Legs

CS106B Summer '21
Assignment 1
Jin-Hee Lee & Grant Bishko

Real photo, taken January 2021 by a real person.



You ~~can't~~ can ~~sit~~ go to YEAH with us!

- Regina George

What *is* YEAH???

`char` the Charmander

Your **E**arly **A**ssignment **H**elp

- Intended to help break down the assignment
- Better understand the *what* and the *how* of what you'll be implementing
- Share key insights to help you!

Word on the street is that YEAH is very helpful even if you don't start early, but we recommend that you do!

More About YEAH:

Live sessions will be held on Wednesdays, 3pm PDT

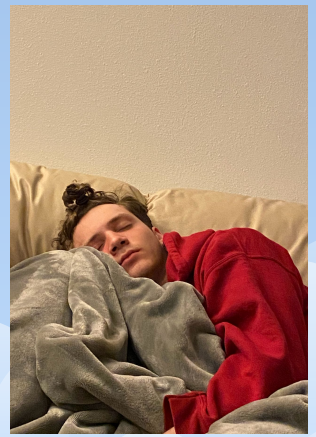
- Come join us TODAY right after lecture!

Will be recorded for those who can't make the time, but...

Join us live!

With students at the live session:

Without students at the live session :(((



Hello world!

Jin-Hee!!

Symbolic Systems, Music

Loves to sing and consume boba

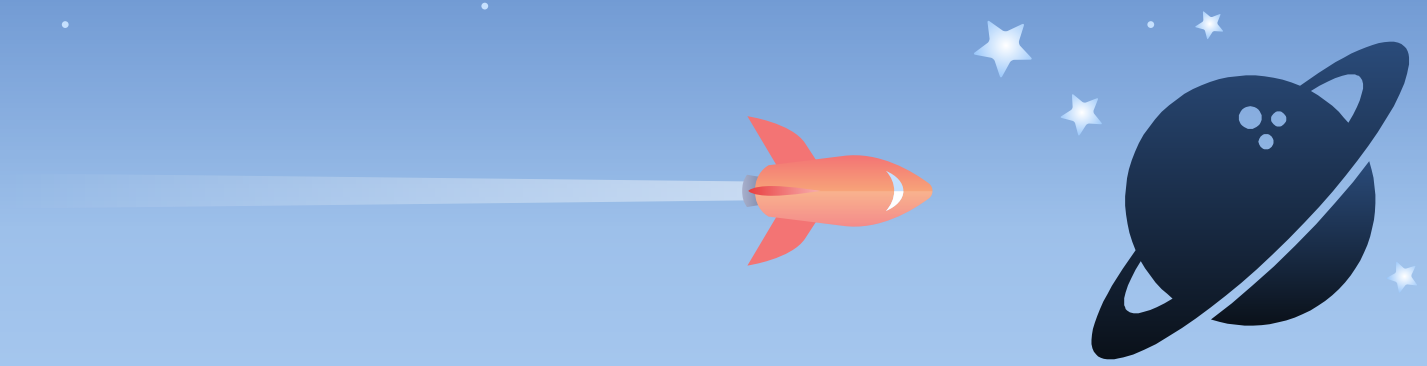
Grant!!

Computer Science?? Music?? Symbolic Systems?? Stay tuned! Grant would also like to know!

Assignment 1 ~Logistics~

- Due Tuesday, June 29 at 11:59pm PDT
  - Turn in on time for a small bonus
  - Penalty-free grace period ends Thursday, July 1 at 11:59pm PDT
- Assignment must be completed **individually**
  - Following the honor code, no shared code

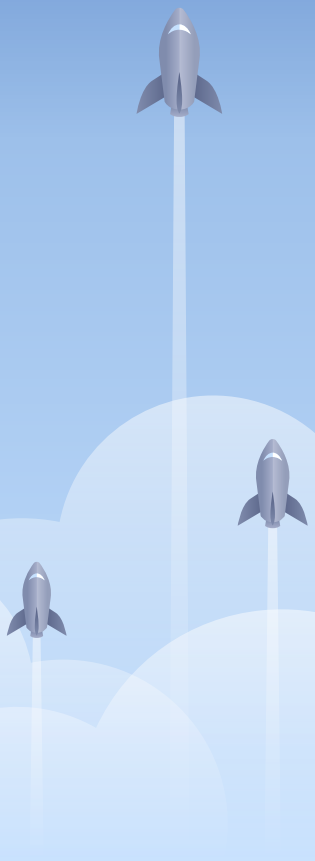

Questions about logistics?



Off we go to get our C++ legs!

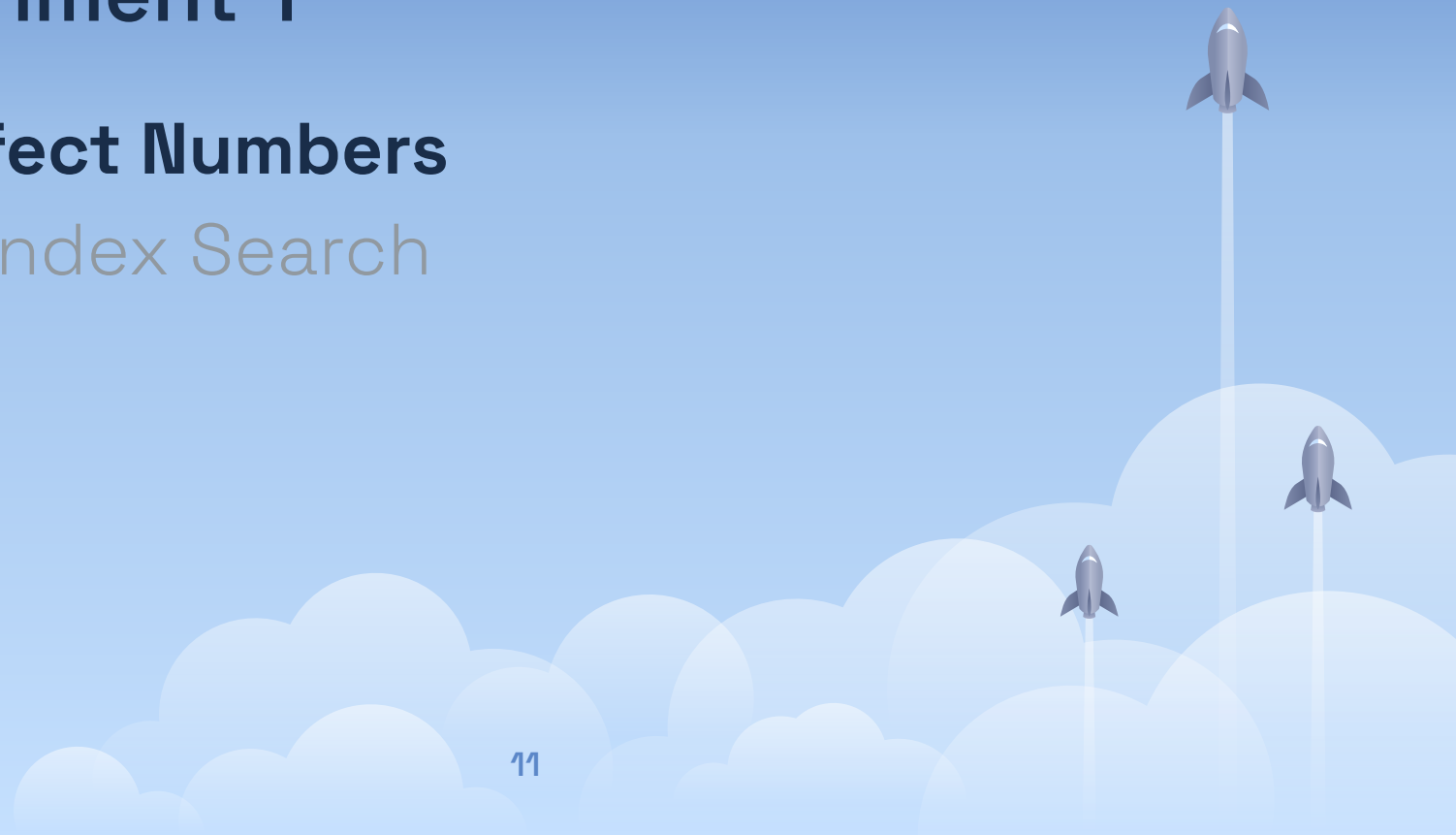
Assignment 1

1. Perfect Numbers
2. Soundex Search



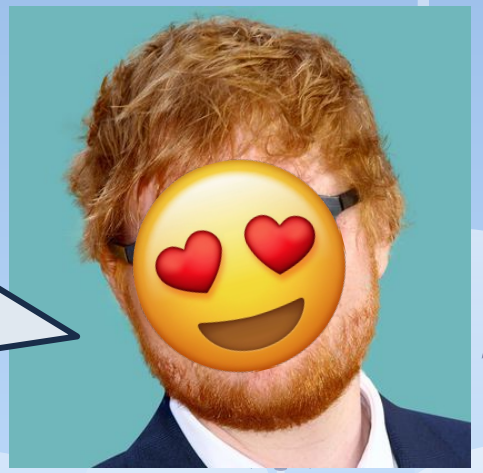
Assignment 1

1. **Perfect Numbers**
2. Soundex Search



Part 1: Perfect Numbers

Baby, you look perfect tonight ♫



Perfect Number

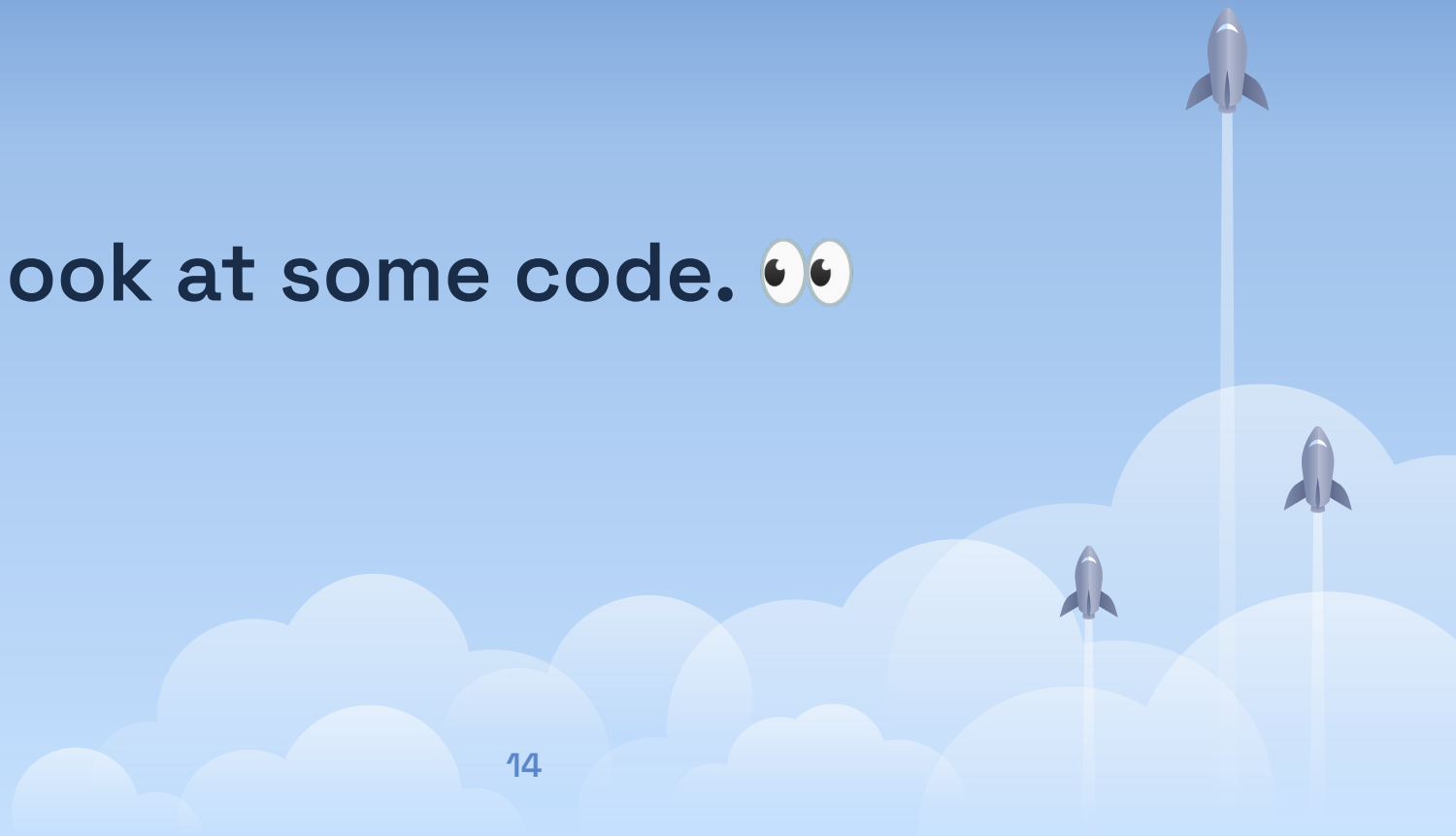
A *perfect number* is an integer that is equal to the sum of its proper divisors.

*Note: we do NOT include the integer itself.

6 = 1 + 2 + 3

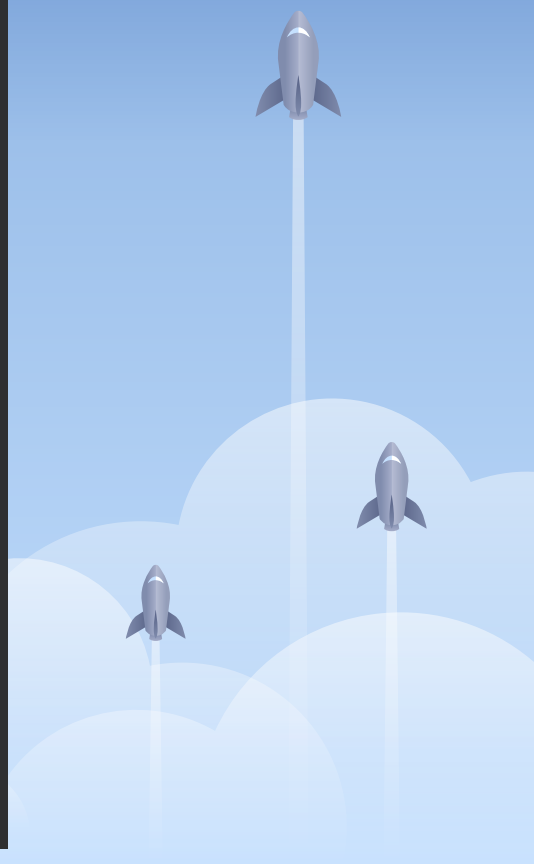
28 = 1 + 2 + 4 + 7 + 14

Let’s look at some code. 👀



divisorSum()

```
12  /* This function takes one argument `n` and calculates the sum ...*/
21  long divisorSum(long n) {
22      long total = 0;
23      for (long divisor = 1; divisor < n; divisor++) {
24          if (n % divisor == 0) {
25              total += divisor;
26          }
27      }
28      return total;
29  }
30
31  /* This function takes one argument `n` and returns a boolean ...*/
36  bool isPerfect(long n) {
37      return (n != 0) && (n == divisorSum(n));
38  }
39
40  /* This function does an exhaustive search for perfect numbers. ...*/
45  void findPerfects(long stop) {
46      for (long num = 1; num < stop; num++) {
47          if (isPerfect(num)) {
48              cout << "Found perfect number: " << num << endl;
49          }
50          if (num % 10000 == 0) cout << "." << flush; // progress bar
51      }
52      cout << endl << "Done searching up to " << stop << endl;
53  }
```



divisorSum()

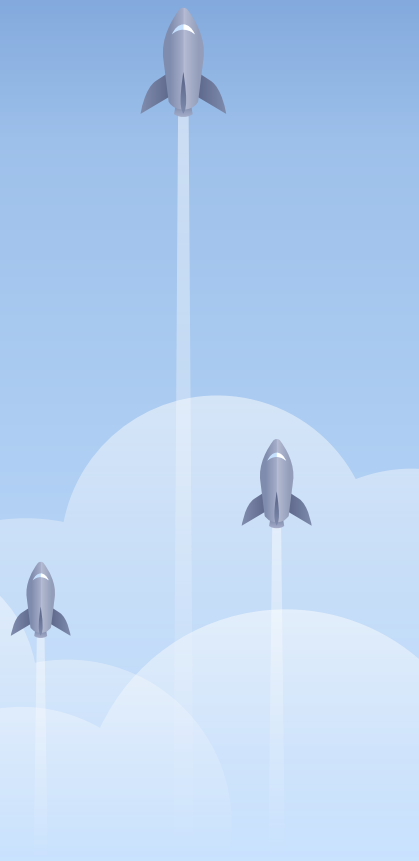
```
12  /* This function takes one argument `n` and calculates the sum ...*/
21  long divisorSum(long n) {
22      long total = 0;
23      for (long divisor = 1; divisor < n; divisor++) {
24          if (n % divisor == 0) {
25              total += divisor;
26          }
27      }
28      return total;
29  }
```

divisorSum()

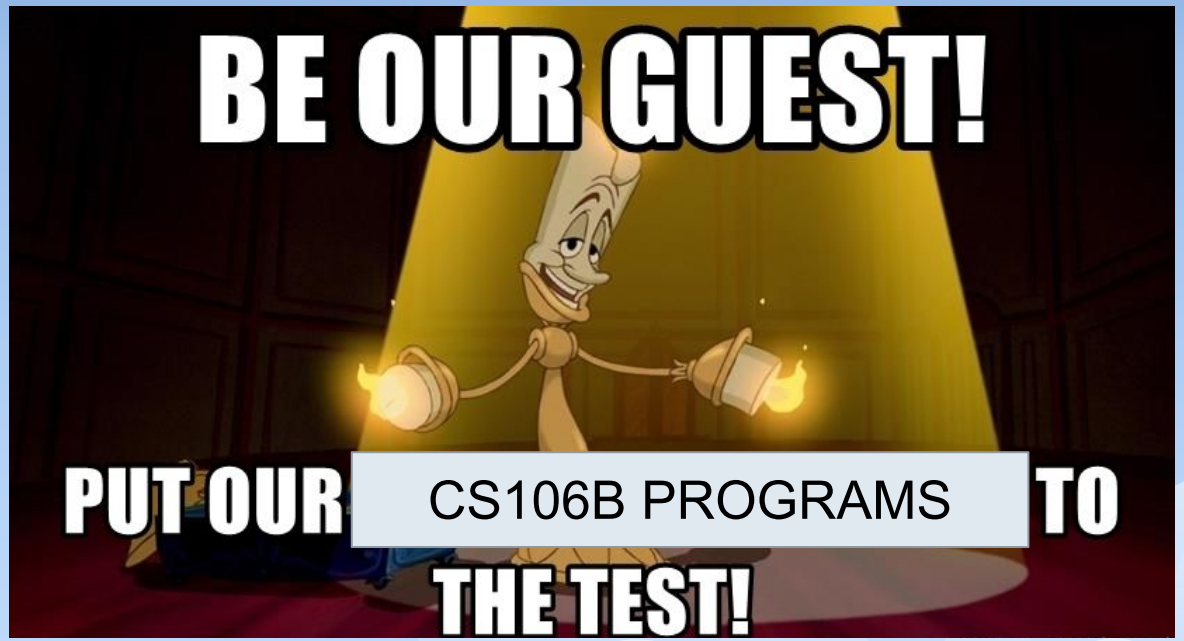
```
12  /* This function takes one argument `n` and calculates the sum ...*/
21  long divisorSum(long n) {
22      long total = 0;
23      for (long divisor = 1; divisor < n; divisor++) {
24          if (n % divisor == 0) {
25              total += divisor;
26          }
27      }
28      return total;
29  }
```

Exhaustive Algorithm

- An algorithm that tries all possible options to find the desired information.
- `divisorSum()` checks every integer starting from `1` to `n`, checking each integer to see if it's perfect.
- We'll test how long this takes...



Testing is the key to success!



Testing using SimpleTest

We'll focus on 4 types of tests:

`EXPECT` tests that the provided predicate is true.

`EXPECT_EQUAL` tests that the 2 provided arguments are equal.

`EXPECT_ERROR` tests that, indeed, an error occurs when running the provided code.

`TIME_OPERATION` times how long the provided code takes to execute.

Trip's Example:


```
int returnFive (int num) {
    if (num == -1) error ("Error, cannot process -1!");
    return 5;
}
// EXPECT tests the provided predicate.
STUDENT_TEST ("Verifies that returnFive returns five with EXPECT") {
    EXPECT (returnFive (0) == 5);
}
// EXPECT_EQUAL compares two values.
STUDENT_TEST ("Verifies that returnFive returns five with EXPECT_EQUAL") {
    EXPECT_EQUAL (returnFive (0), 5);
}
// EXPECT_ERROR passes if and only if the code it runs throws an error.
STUDENT_TEST ("Verifies that returnFive throws an error on bad input with EXPECT_ERROR") {
    EXPECT_ERROR (returnFive (-1));
}
```


Test example program written by Trip Master.

TIME_OPERATION

- Takes 2 parameters: `inputsize`, `operation`
  - Times and shows how long it takes to perform the `operation` on `inputsize` elements.


Questions about testing in 106B?

Level Up: let’s do better!

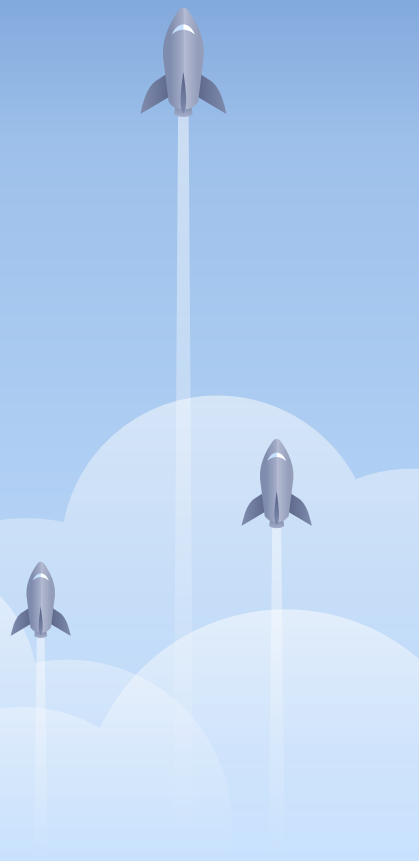
Example time!

Let's unpack `isPerfect(36)`

Divisors of 36: 1, 2, 3, 4, 6, 9, 12, 18

Right now, the algorithm has to loop through all numbers from 1 through (36 - 1).

But let's take a closer look at the divisors:



Example time!

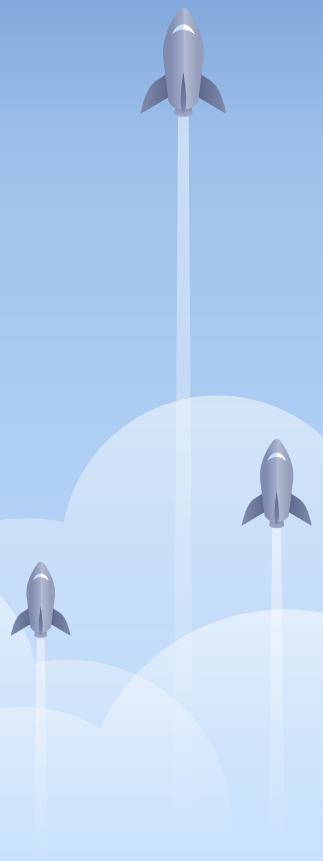
Let's unpack `isPerfect(36)`

Divisors of 36: 1, **2**, 3, 4, 6, 9, 12, **18**

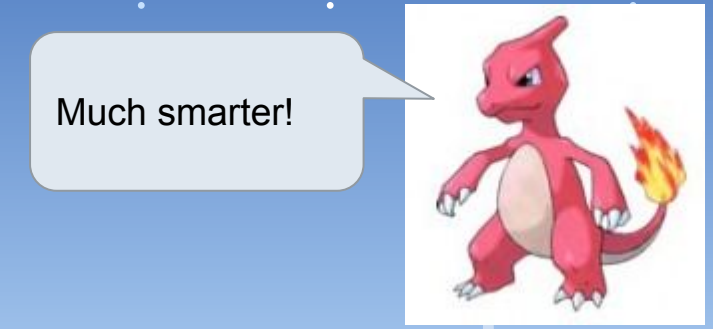
1, 2, **3**, 4, 6, 9, **12**, 18

1, 2, 3, **4**, 6, **9**, 12, 18

1, 2, 3, 4, **6**, 9, 12, 18



Example time!



Let's unpack `isPerfect(36)`

Divisors of 36: 1, 2, 3, 4, 6, 9, 12, 18

Since each divisor has a corresponding term that multiplies to the product, we only have to loop from 1 through `sqrt(36)`!

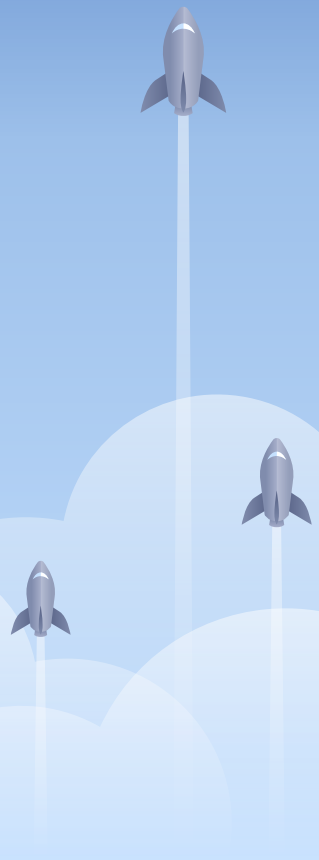
- [1], [2 and corresponding term], [3 and corresponding term] ...

smarterSum()

Your task: write `long smarterSum(long n)`

Tips:

- Use divisorSum() as a starting point
- Consider your **edge cases**

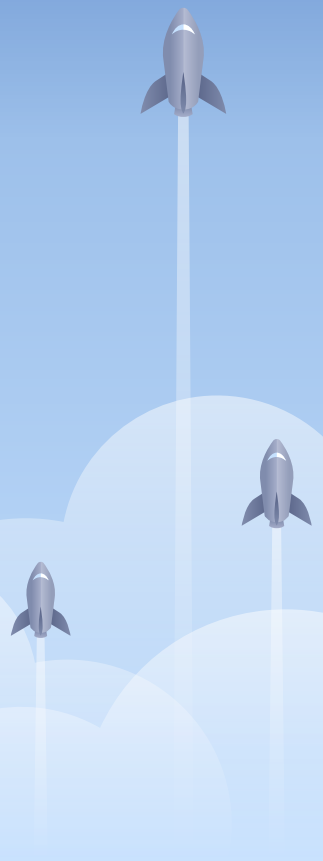


What is an edge case?

An edge case occurs when working with an extreme parameter.

For example, if you're working with `int`s, your program should be able to handle oddball inputs like `0, 1, -1`.

With `string`s, a standard edge case is `""`.

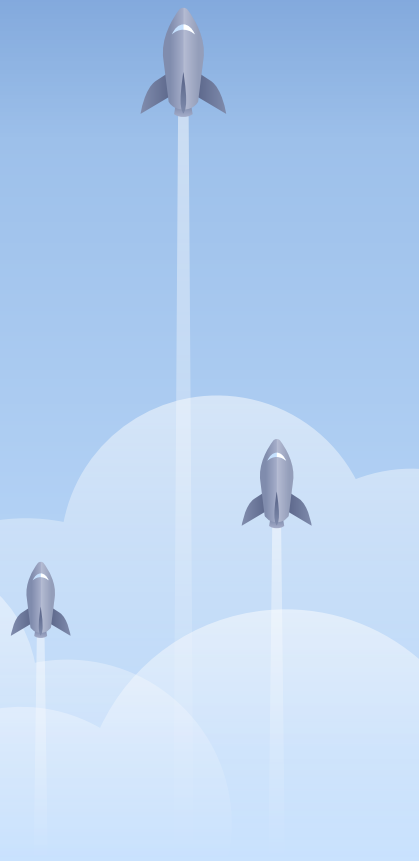


smarterSum()

Your task: write `long smarterSum(long n)`

Tips:

- Use divisorSum() as a starting point
- Consider your **edge cases**
  - Typical int edge cases (previous slide)
  - Is there anything special about the square root?
    - If so, how should we handle it?

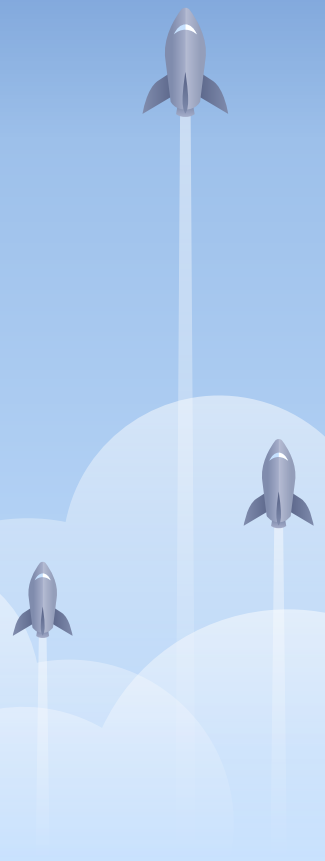



Questions about smarterSum()?

Level Up: even stronger!

Mersenne Prime

- A prime number is one whose only divisors are 1 and the number itself.
- A Mersenne prime is a prime number that is one less than a power of two.
- 2^n - 1 for some integer n
- Example: **31** is prime and can be expressed as **2^5 - 1**. 31 is a Mersenne prime.



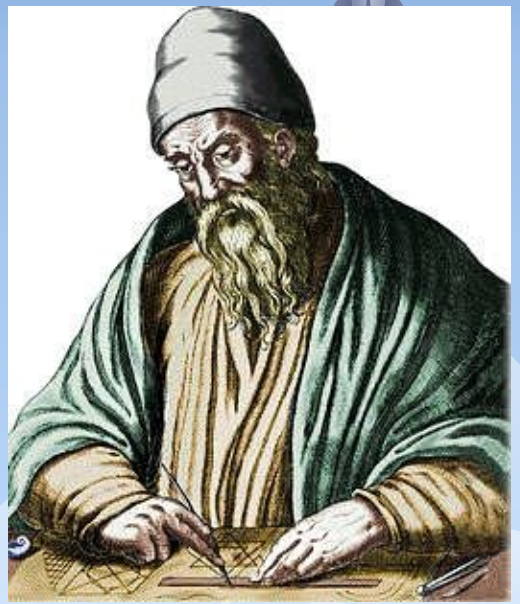
Enter Euclid...

He discovered a cool relationship:

If 2^k - 1 is prime, then 2^(k-1)*(2^k - 1) is a perfect number!

With this in mind...

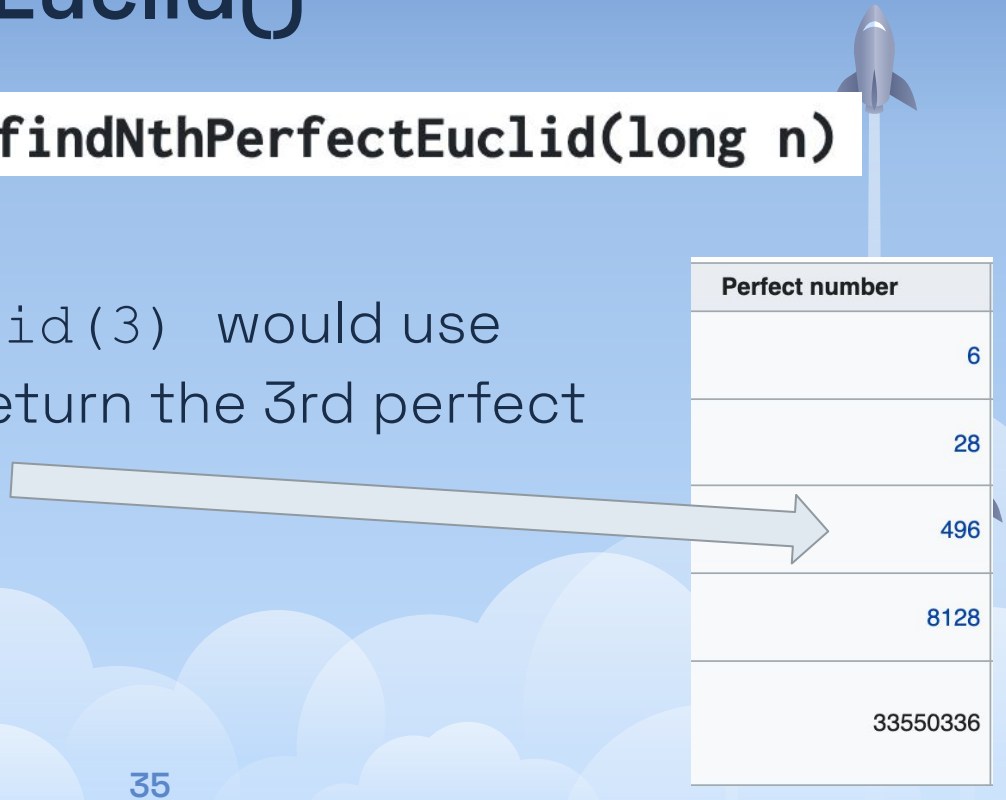
Here is Euclid, helping us write the first 106B assignment. (300 BC)



findNthPerfectEuclid()

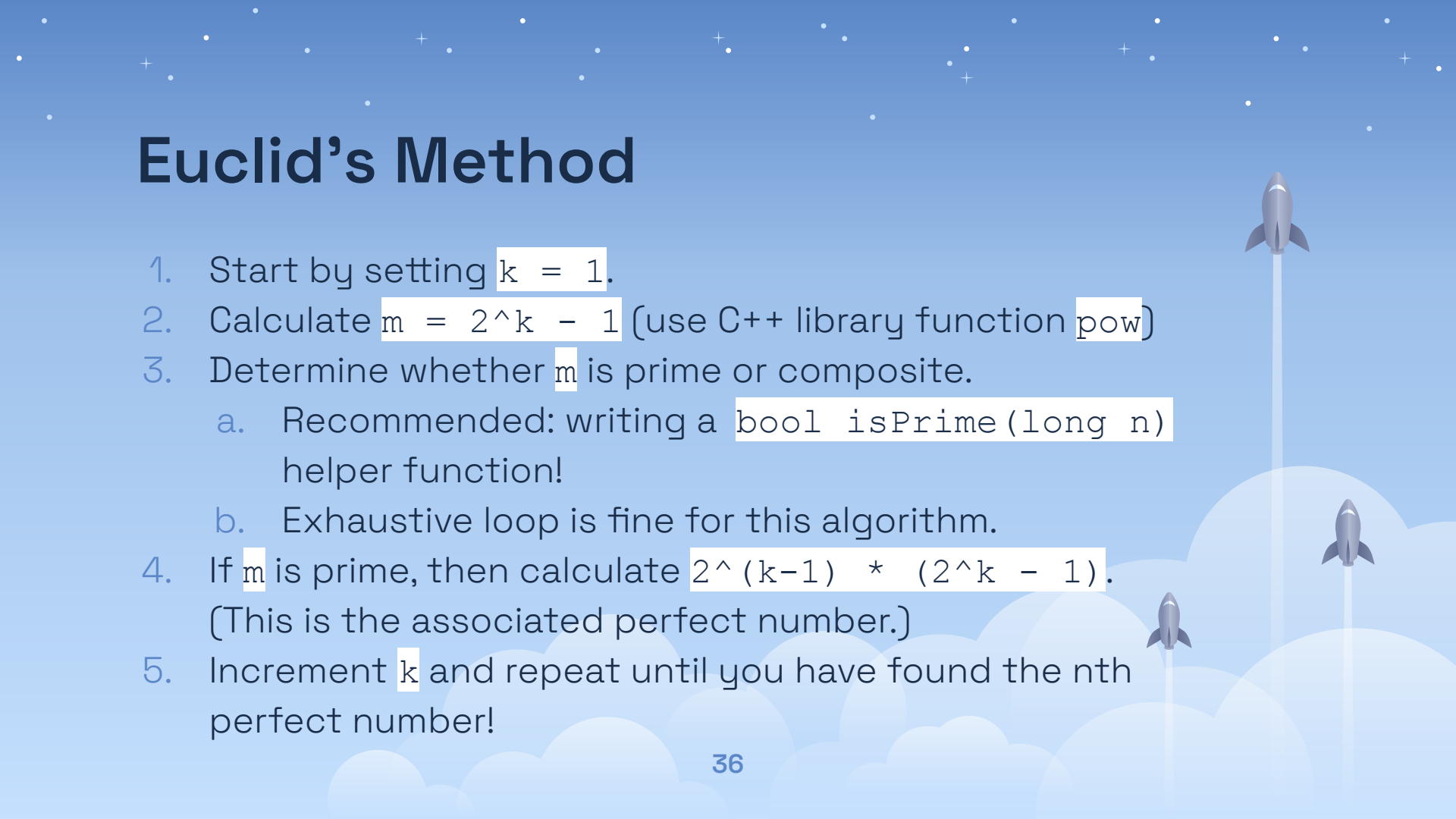
Your task: write `long findNthPerfectEuclid(long n)`

For example, the call `findNthPerfectEuclid(3)` would use Euclid's method and return the 3rd perfect number, which is 496.



Euclid's Method

1. Start by setting `k = 1`.
2. Calculate `m = 2^k - 1` (use C++ library function `pow`)
3. Determine whether `m` is prime or composite.
    a. Recommended: writing a `bool isPrime(long n)` helper function!
    b. Exhaustive loop is fine for this algorithm.
4. If `m` is prime, then calculate `2^(k-1) * (2^k - 1)`. (This is the associated perfect number.)
5. Increment `k` and repeat until you have found the nth perfect number!

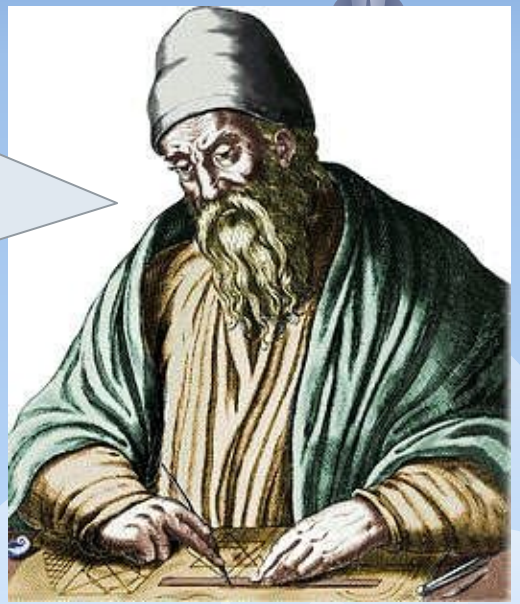


Some comments from Euclid:

We are using `long` instead of `int` since these numbers can get really big!

If you're on a Windows machine, you should be able to find around 6 perfect numbers. On a Mac, you should be able to find a few more!

- If you're interested, you can look up 32-bit systems vs. 64-bit systems. Also, take CS107 :)



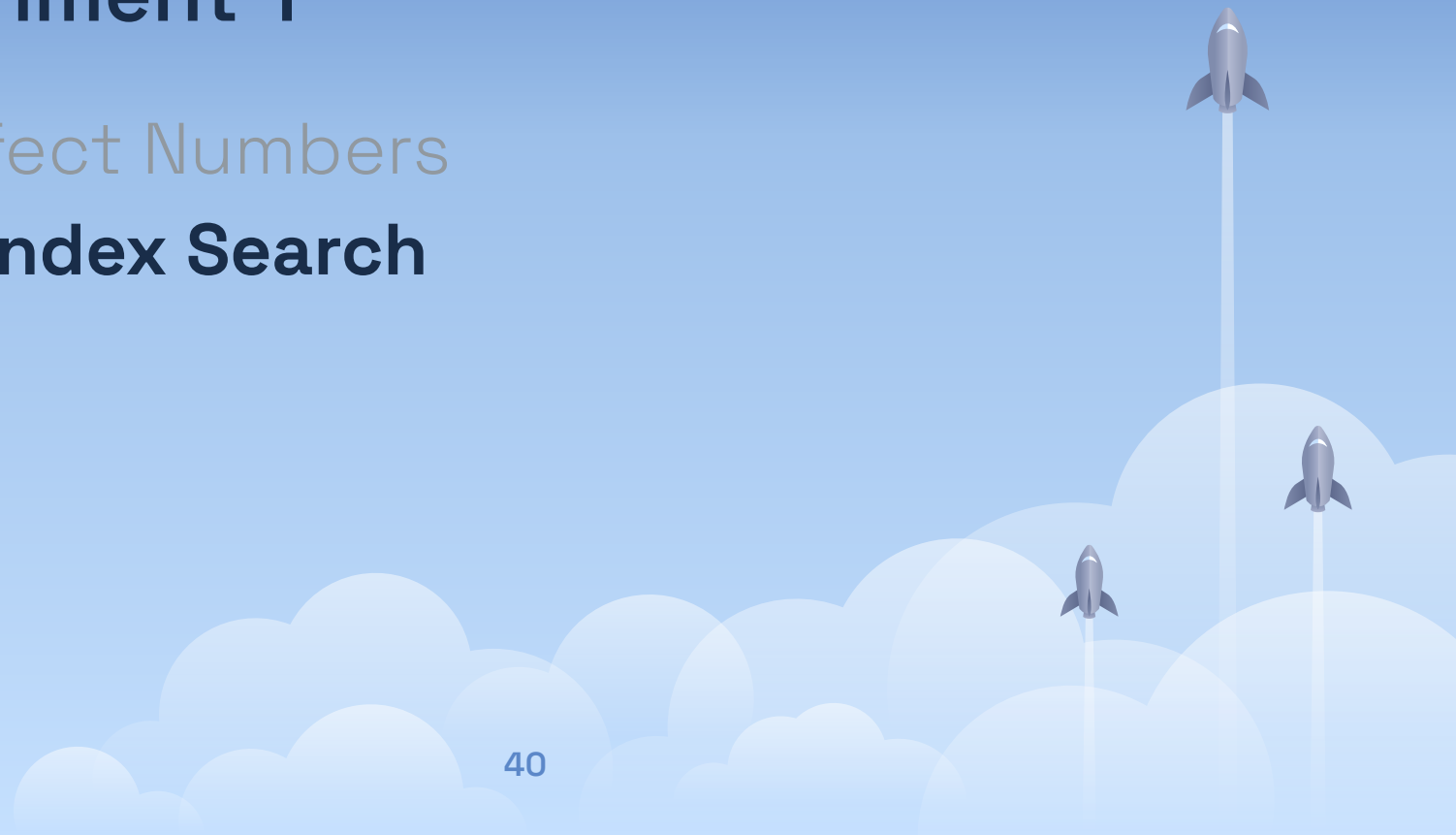

Questions about
findNthPerfectEuclid()?

Let’s breathe.

That was a lot of information!

Assignment 1

1. Perfect Numbers
2. **Soundex Search**

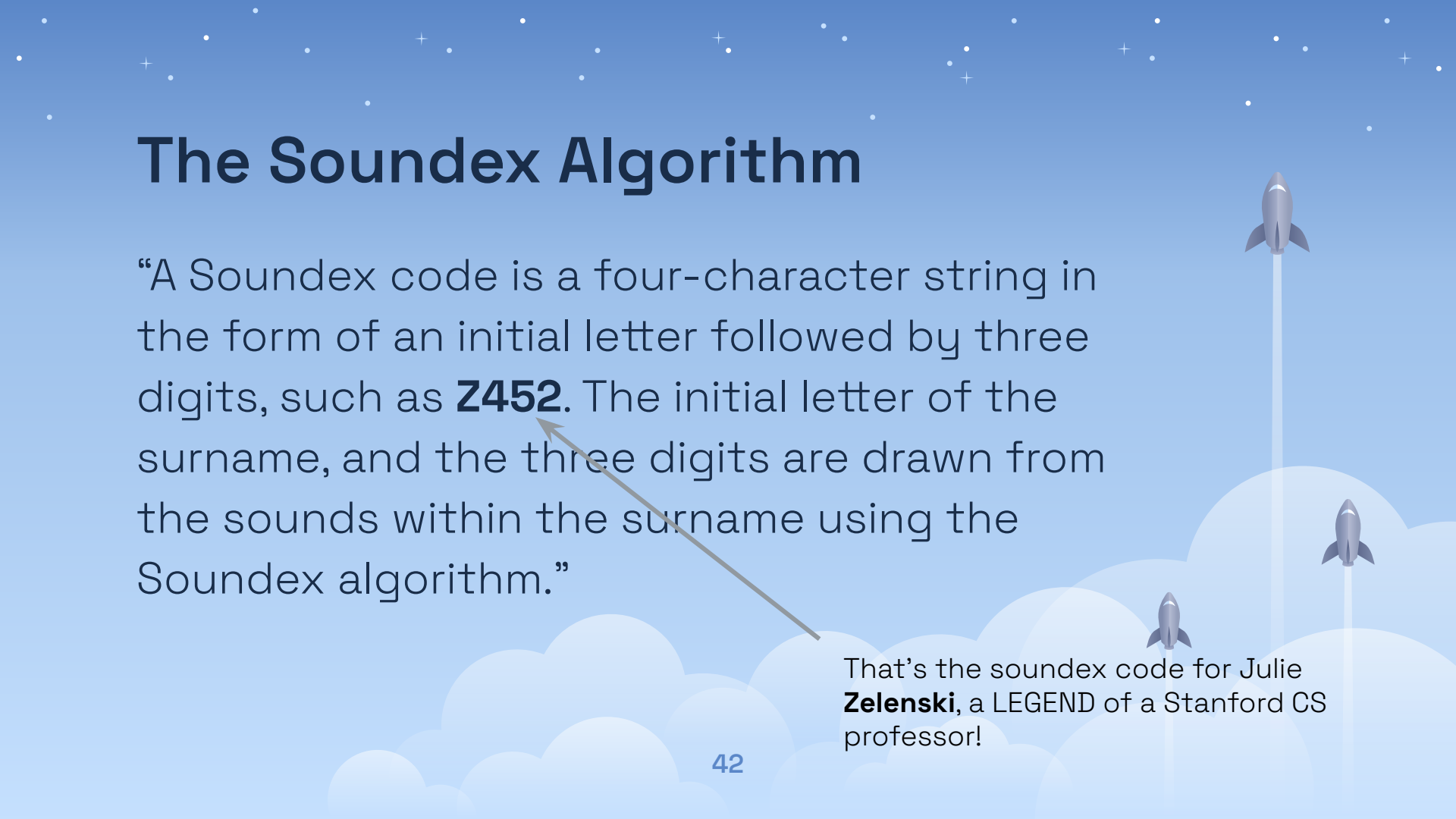


Part 2a: Soundex()

- You will be writing a program that **takes in a last name** and turns it into a **soundex code**
  - Essentially a 4-digit pseudo-phonetic representation of a last name
- Fun fact! The US census uses soundex codes!

The Soundex Algorithm

“A Soundex code is a four-character string in the form of an initial letter followed by three digits, such as **Z452**. The initial letter of the surname, and the three digits are drawn from the sounds within the surname using the Soundex algorithm.”



That’s the soundex code for Julie **Zelenski**, a LEGEND of a Stanford CS professor!

The Soundex Algorithm

Lucky for us, the actual steps to turn a last name into a soundex code is pretty straightforward!

Let’s take a look!

The Soundex Algorithm

1. Discard all non-letters from the surname (dashes, spaces, apostrophes, etc.)
2. Encode each letter as a digit from the table to the right
3. Coalesce adjacent duplicate digits from code (222025 becomes 2025)
4. Replace first digit of code with first letter of the original surname, converting to uppercase
5. Remove all zeros from the code
6. Make the code exactly 4 digits by padding with zeros or truncating the excess

Digit	represents the letters
0	A E I O U H W Y
1	B F P V
2	C G J K Q S X Z
3	D T
4	L
5	M N
6	R


Let’s do one together!

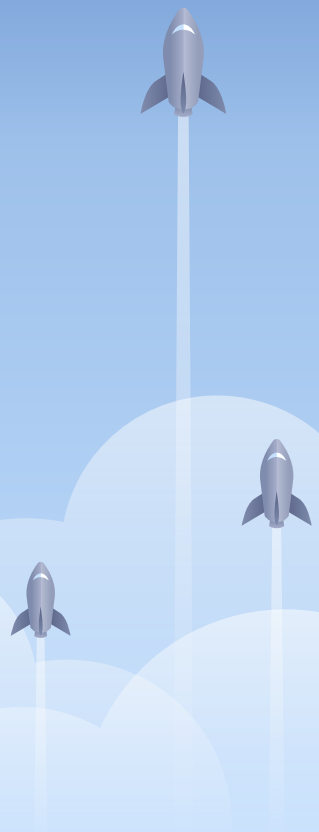
Example Surname: ***Master***

Step 1: Discard all non-letters from the surname (dashes, spaces, apostrophes, etc.)

Hint: use the **isAlpha()** function here!

Master → Master

(There are no non-letters!)



Example Surname: ***Master***

Step 2: Encode each letter as a digit from the table

Master → **502306**

Digit	represents the letters
0	A E I O U H W Y
1	B F P V
2	C G J K Q S X Z
3	D T
4	L
5	M N
6	R

Example Surname: ***Master***

Step 3: Coalesce adjacent duplicate digits from code (222025 becomes 2025)

502306 → 502306

(No duplicate digits)

Example Surname: ***Master***

Step 4: Replace first digit of code with first letter of the original surname, converting to uppercase

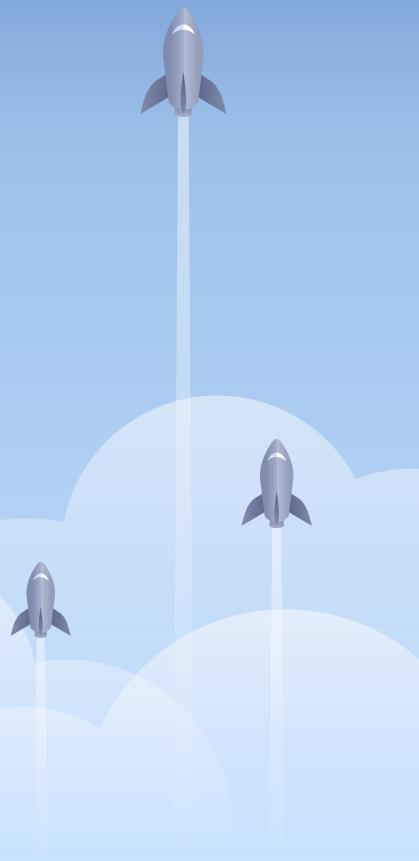
502306 → **M02306**

Hint: Save the first character of the surname right after step 1! (Here, we'd save 'M')

Example Surname: ***Master***

Step 5: Remove all zeros from the code

M02306 → **M236**



Example Surname: ***Master***

Step 6: Make the code exactly 4 digits by padding with zeros or truncating the excess

M236

(already 4 digits!)

We are done!

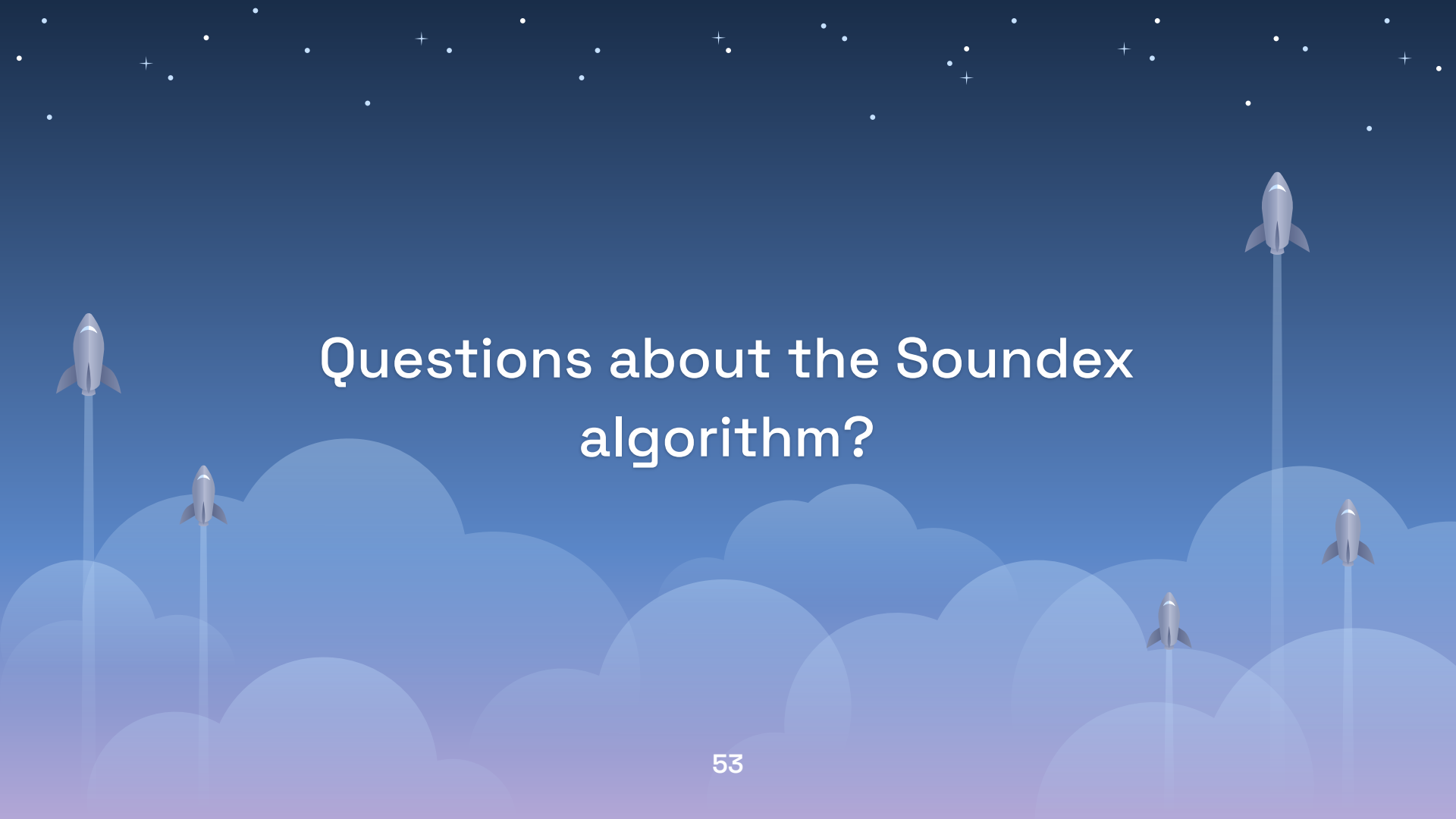
Example Surname: ***Master***

Step 6: Make the code exactly 4 digits by padding with zeros or truncating the excess

If we do need to pad/truncate, here are some examples of what that would look like:

M2 → M200

M23613 → M236

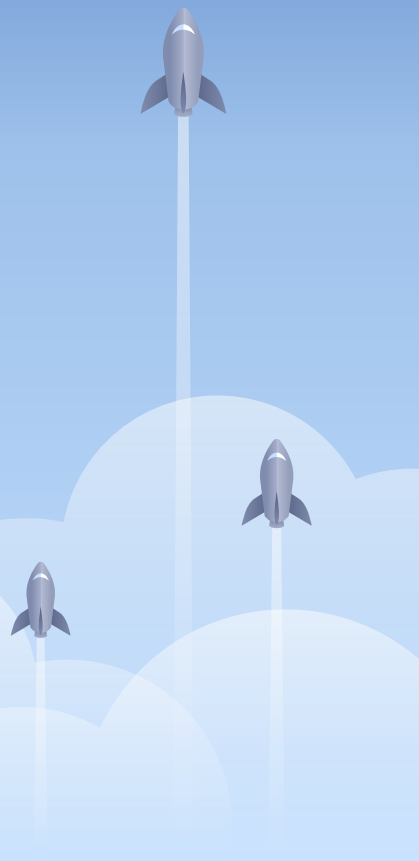

Questions about the Soundex algorithm?

Short Answers!

As you complete the assignment, you will be asked short answer questions.

These consist of things like:

- Calculate the soundex code for ____
- What is the soundex code for your own last name?
- Game plan for writing the code!



DECOMPOSE!!!!

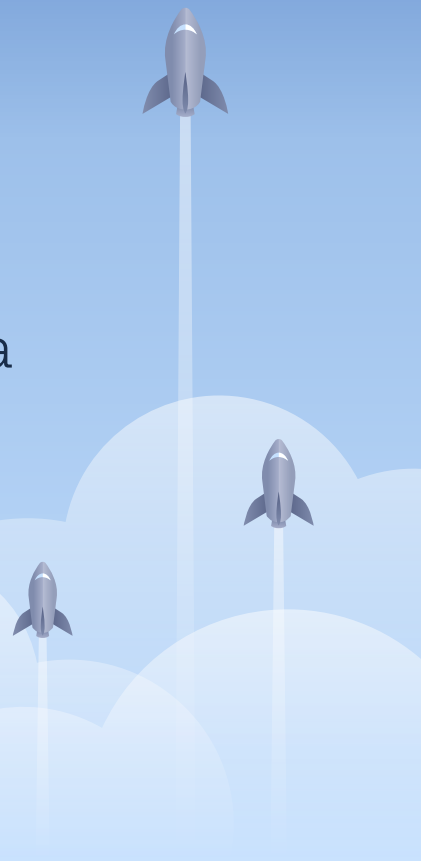
You will be writing the code for the function:

```
string soundex(string s)
```

Which takes in a string (surname) and returns a string (the soundex code).

But, calculating the soundex code for a name is **complicated**! We STRONGLY RECOMMEND DECOMPOSING this function into many helpers!

Hint: make a helper for each step of the process

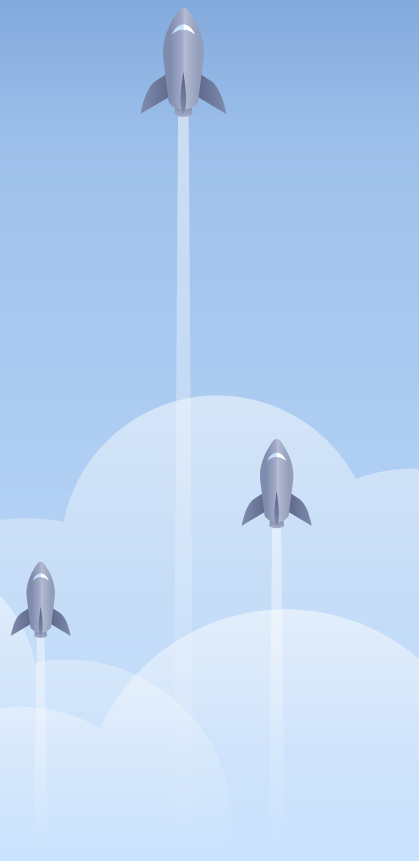


Why is decomposing helpful?

TESTING TESTING TESTING

If you decompose your code into many functions, you will be able to test as you go, and pinpoint any bugs to a specific step along the soundex-encoding process!

Write tests for each helper function you create!

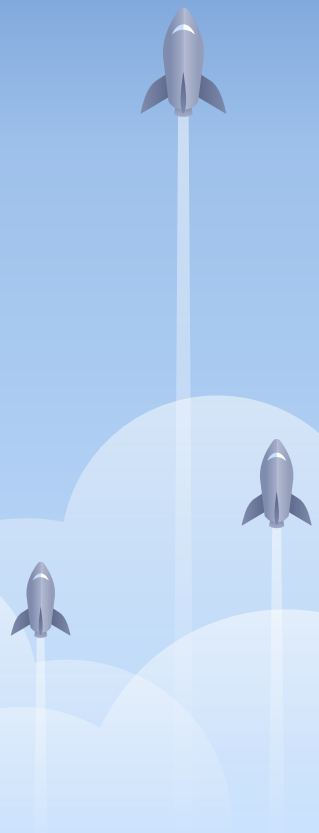


Why is decomposing helpful?

They give you a buggy version of a pre-decomposed function for you to fix!

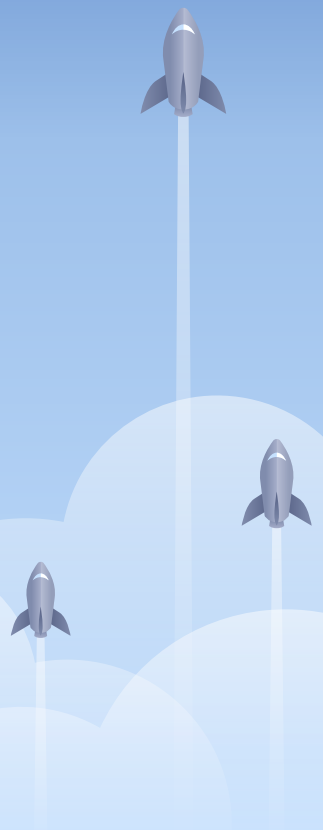
(removeNonLetters)

See how helper functions make finding these bugs much easier?



Tips for part 2a: soundex()

- Lots of string/char work here!
  - Note that in C++ a string is represented by “” while a char uses ‘
  - Indexing into a string is going to be super helpful here (like using `str[i]`)
  - In case you need to convert between char/string, strlib.h has some helpful functions!



Tips for part 2a: soundex()

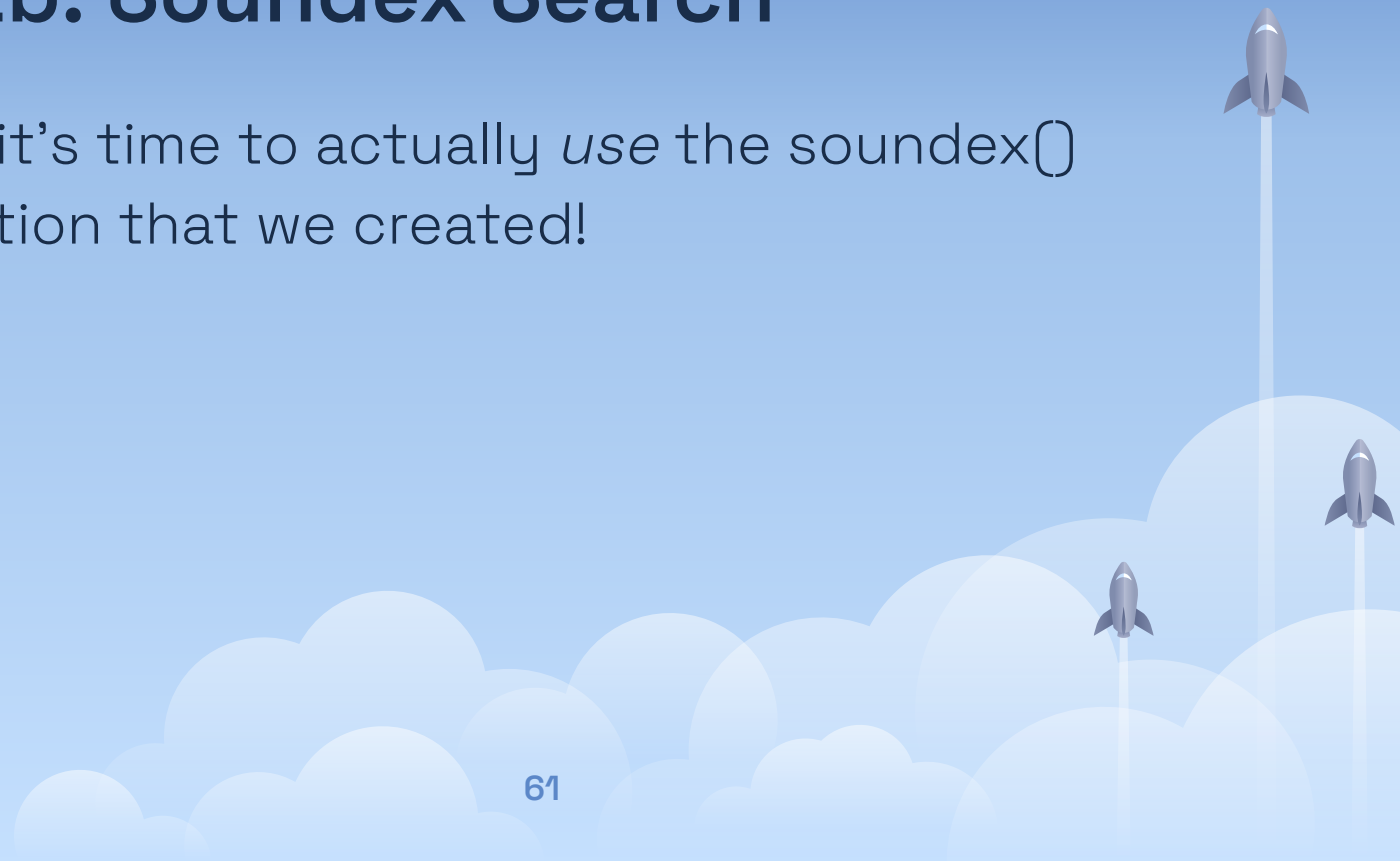
- Make sure your code is **case insensitive**, so "Zelenski" → Z452 and "zElenSkI" → Z452
- Add your own tests
- If you ever need to convert to uppercase, the Stanford library function:
  `toUpperCase(string s)` will do that! So will the standard C++ function
  `toupper(int c)`

Takes in an integer?? Why?? ASCII !!! (a numeric representation of characters) We will learn much more about it throughout this course!


Questions about Soundex?

Part 2b: Soundex Search

- Now it's time to actually *use* the soundex() function that we created!

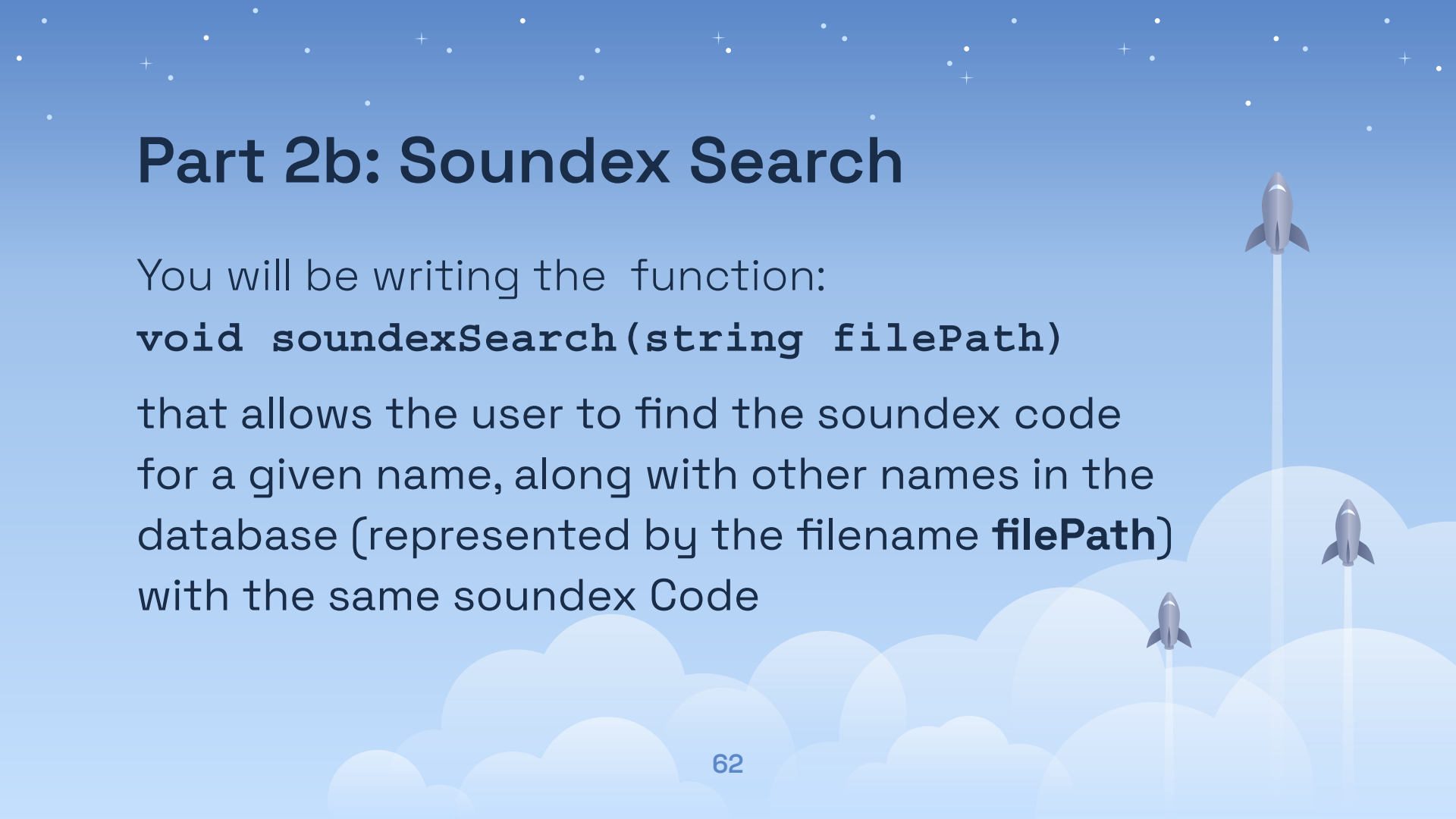


Part 2b: Soundex Search

You will be writing the function:

`void soundexSearch(string filePath)`

that allows the user to find the soundex code for a given name, along with other names in the database (represented by the filename **filePath**) with the same soundex Code



Part 2b: Soundex Search

Match this user interaction exactly!

```
Read file res/surnames.txt, 26584 names found.

Enter a surname (RETURN to quit): Zelenski
Soundex code is Z452
Matches from database: {"Zelenski", "Zelnick", "Zelnik", "Zielonka"}

Enter a surname (RETURN to quit): hanrahan
Soundex code is H565
Matches from database: {"Hammerman", "Hamren", "Haner-McAllister", "Hanrahan"}

Enter a surname (RETURN to quit):
All done!
```

This top line is done for you!

Part 2b: Soundex Search

You have been given the code that takes in a **`filePath`** and reads it into a vector

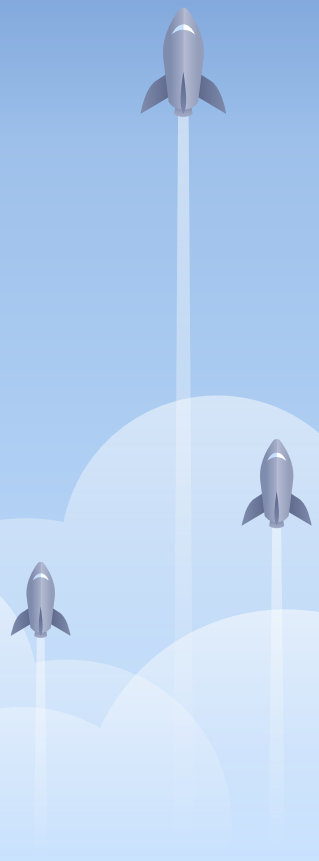
- A Vector is just like a List (in Python) or an ArrayList (in Java).

```
void soundexSearch(string filepath)
{
    // The proivded code opens the file with the given name
    // and then reads the lines of that file into a vector.
    ifstream in;
    Vector<string> lines;

    if (openFile(in, filepath)) {
        readEntireFile(in, lines);
    }
    cout << "Read file " << filepath << ", " << lines.size() << " names found." << endl;
```

Part 2b: Soundex Search

Here's what you will need to do (1/3):

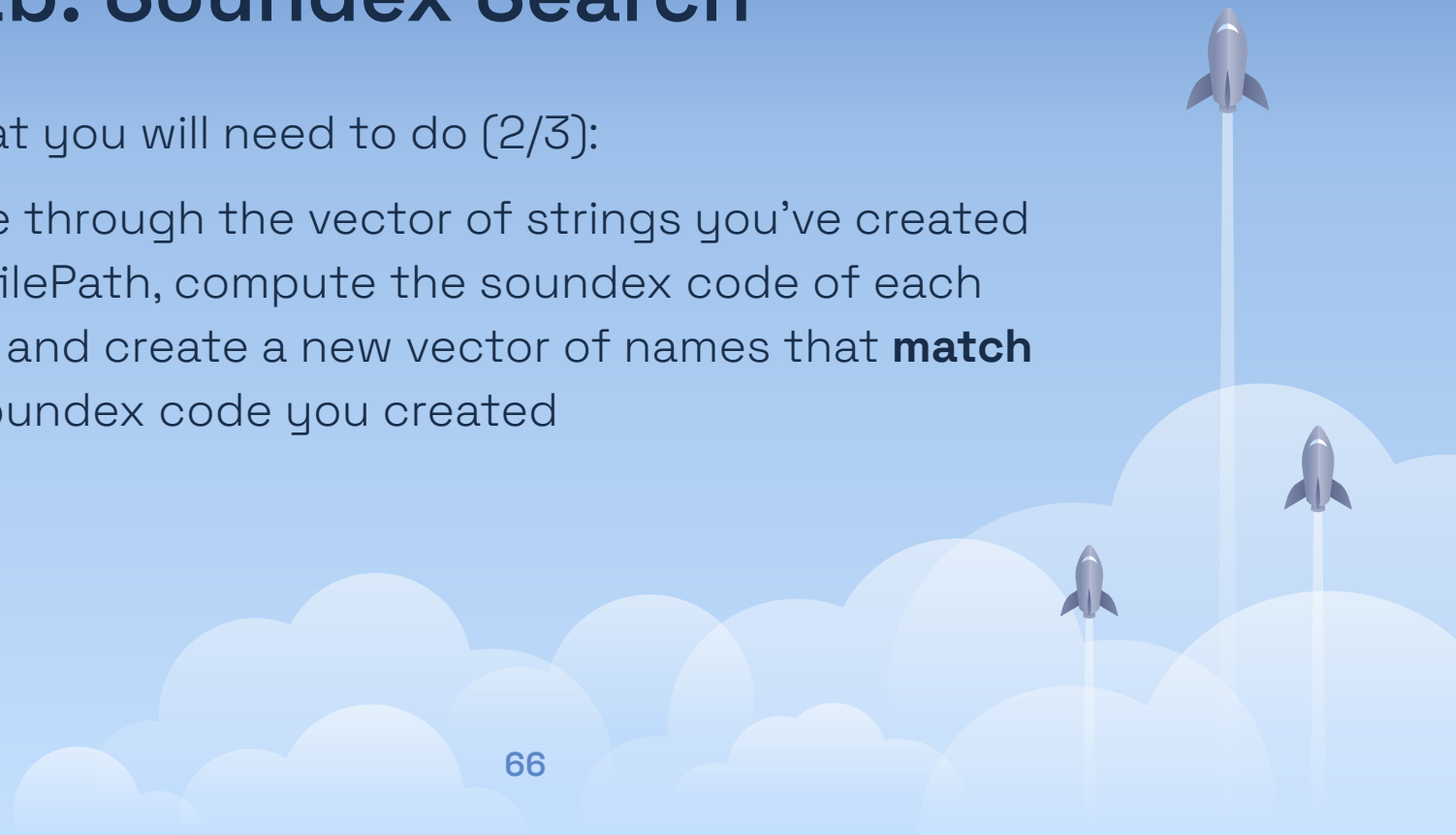
- Repeatedly prompt the user to enter a surname (while loop???) (the function **`getLine`** from "simpio.h" will be super helpful here)
- Compute and print the soundex code for the inputted surname



Part 2b: Soundex Search

Here's what you will need to do (2/3):

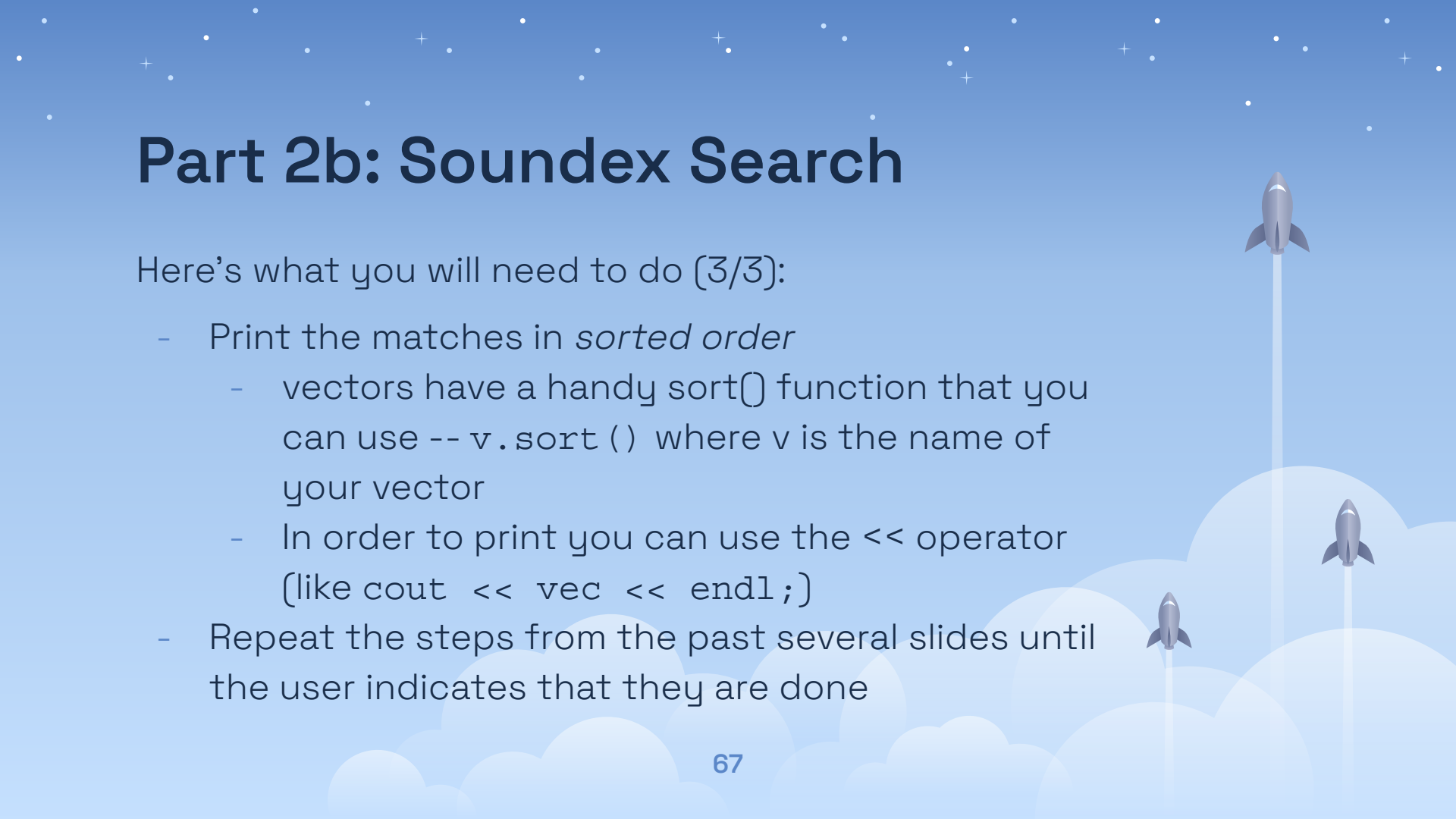
- Iterate through the vector of strings you've created from filePath, compute the soundex code of each name, and create a new vector of names that **match** the soundex code you created



Part 2b: Soundex Search

Here's what you will need to do (3/3):

- Print the matches in *sorted order*
  - vectors have a handy sort() function that you can use -- `v.sort()` where v is the name of your vector
  - In order to print you can use the << operator (like `cout << vec << endl;`)
- Repeat the steps from the past several slides until the user indicates that they are done



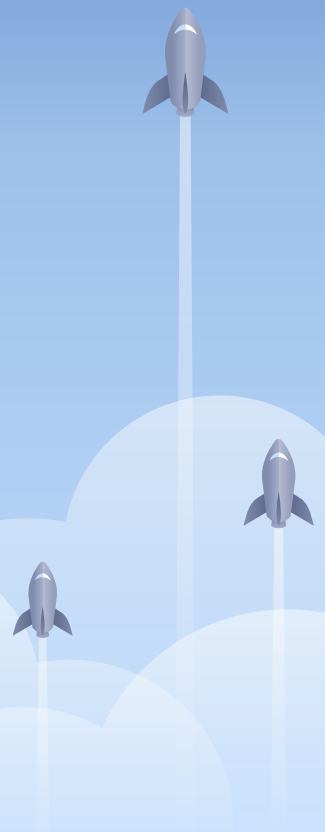
This is what it will look like!

```
Read file res/surnames.txt, 26584 names found.

Enter a surname (RETURN to quit): Zelenski
Soundex code is Z452
Matches from database: {"Zelenski", "Zelnick", "Zelnik", "Zielonka"}

Enter a surname (RETURN to quit): hanrahan
Soundex code is H565
Matches from database: {"Hammerman", "Hamren", "Haner-McAllister", "Hanrahan"}

Enter a surname (RETURN to quit):
All done!
```

Tips for dealing with vectors!

```
Vector<string> names;

// Append to a vector
names += "Trip";
string str = "Kylie";
names.add(str);

// index-based for loop!
for (int i = 0; i < names.size(); i++) {
    string name = names [i];
}

// for-each loop!
for (string name : names) {

}
```

Stolen from Trip Master himself!



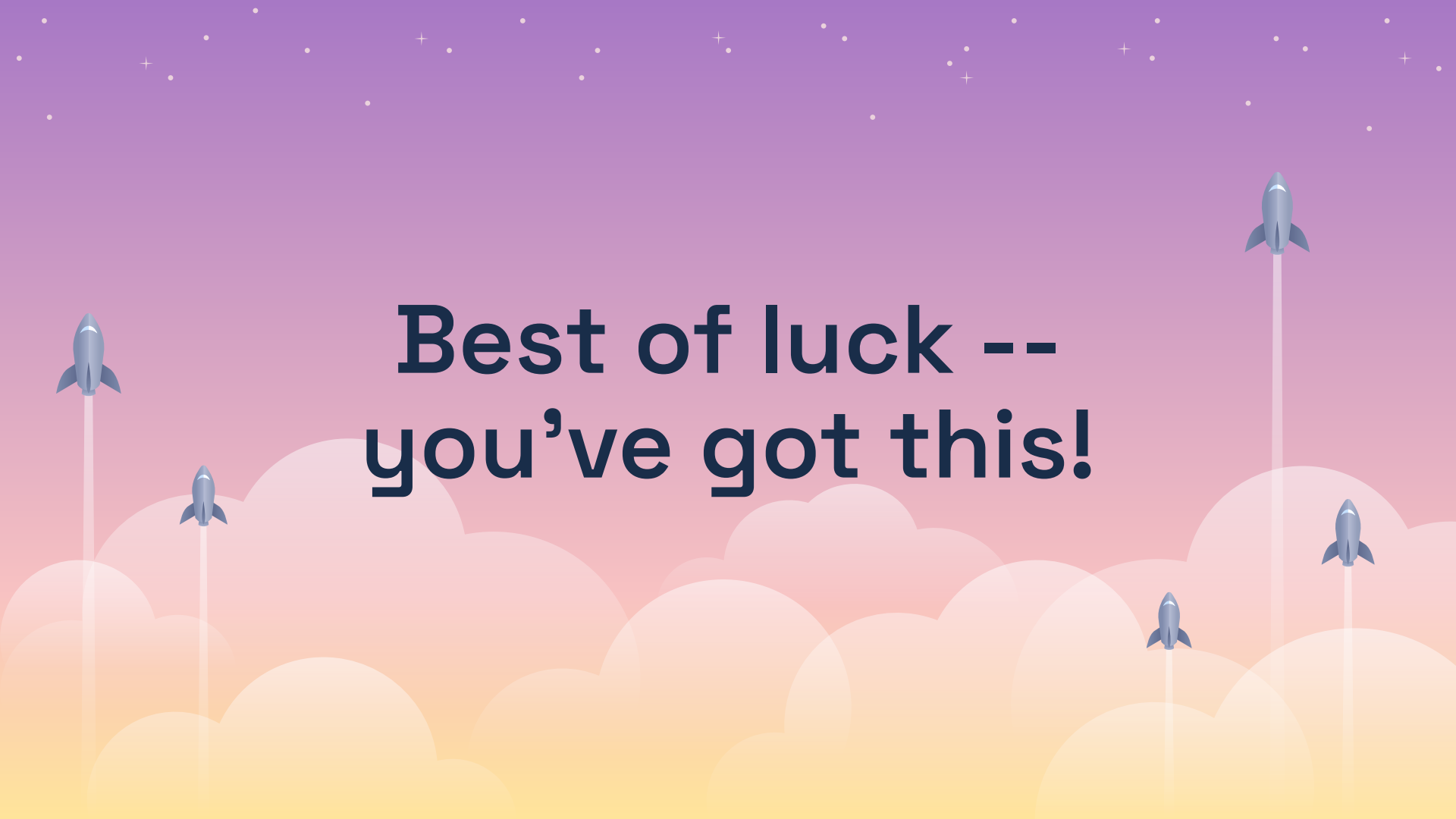
Soundex Search Tips/Tricks :)

- Use getLine (from “simpio.h”) to get user input!
- To sort the names in the vector, just use vectorName.sort()! No need to make things super complicated for yourself
- You can cout (print) vectors just like you would strings, and they will print just like you saw in the example!
- Feel free to Google “Stanford String cpp” to find a bunch of helpful functions you can use!




Questions about
Soundex / Soundex Search?


Final questions?


Best of luck --
you've got this!